Software Engineering Research/Developer Collaborations in 2004 (CI04)

Final Report
(External Release)

April 26, 2005

Tom Pressburger, ARC
Lawrence Markosian, QSS Group Inc. (ARC)

Table of Contents

1 Overview

In 2004, six collaborations between software engineering technology providers and NASA software development personnel deployed a total of five software engineering technologies (for references, see Section 7.2) on the NASA projects. The main purposes were to benefit the projects, infuse the technologies if beneficial into NASA, and give feedback to the technology providers to improve the technologies. Each collaboration project produced a final report (for references, see Section 7.1). Section 2 of this report summarizes each project, drawing from the final reports and communications with the software developers and technology providers. Section 3 indicates paths to further infusion of the technologies into NASA practice. Section 4 summarizes some technology transfer lessons learned. Section 6 lists the acronyms used in this report.

Below we restate our success criteria from our SARP proposal to oversee the collaborations:

> "We would like one of the main success criteria to be that the research products used in the collaborations are adopted for future software development by the teams (or organization). However, this is unrealistic for mid TRL-level research products that may lack productization, and it may be unrealistic for high TRL or even for commercial products (for example, the license fee may be too high for a single team to bear). Thus we have identified several other success criteria.
>
> 1. The success criteria of the collaboration projects funded under this proposal are met. This includes a positive rating for each product on the evaluation criteria metric.
> 2. The research product is adopted by the collaborating software development team for current use.
> 3. The research product is included in a list of recommended development practices at a NASA Center or by contractor.
> 4. The software development team using the product provides feedback, including performance data, to the research team to guide future development of the product.
> 5. Six months after the funded collaboration period, the research product is still being used by the development project or by a successor development project.
> 6. The researchers and consumers recommend to the CTO methods of making future versions of the research products available within NASA (for example, by Open Sourcing or by licensing the technology commercially or to organizations such as the Southwest Research Institute).
> 7. Independent of the success of the collaborations, "lessons learned" regarding the challenges and success factors for software development technology infusion within NASA."

A modification of 3 is "The research product is recommended for a branch, division, or directorate at a center". That is the statement for which column 3 applies in the table below.

Also relevant to judging the impact of the collaborations is the penetration factor (PF) used for SARP quarterly reviews:

> PF 8: Data passed back to project;
> PF 9: Results actually used by the project.

The following table summarizes the impacts of the technology in each collaboration regarding the penetration factor and success criteria as well as brief notes.

Y = Yes **A** = Anticipated in 2005 – 2006 timeframe

Project	PF	1	2	3	4	5	6	7	Impacts
ARC CGS on ISS payload software	9				Y			Y	Found 2 errors to be fixed. Useful feedback to the CGS developers.
GSFC PBI in Flight Software Branch	9	Y	A	Y	Y	A		Y	PBI led to changes in a project's development plan. Expect roll out of PBI in FSB standards.
JPL ODC on ground software	8 now, will soon be 9	Y	Y	Y	Y	A		Y	Training occurred in several JPL organizations. ODC led to several recommendations that will be used in project maintenance phase. Collaboration is continuing to infuse ODC on another project.
JSC CodeSurfer for Inspections of ISS software	9	Y	Y		Y			Y	Found 12 additional (minor) defects. Tool is continuing to be used and promulgated.
MSFC SWAT & CGS on Flight Software	9	½	Conditional on cost	Conditional on cost	Y			Y	Useful feedback to the CGS developers. SWAT found 9 defects worth fixing in the software, some of which had escaped formal testing.
USA PBI on ISS software	9	Y	Y	Y	Y	A		Y	Found 6 "major" defects, several of which had escaped previous inspections, and/or occurred in reused code. Will continue to be used and will be recommended as an optional process.

2 Summary of technology provider/software development project collaborations

This section describes briefly each collaboration: its objectives, what transpired, its impact on the project, and the success criteria that were met.

2.1 ARC: "On Orbit Software Analysis" using C Global Surveyor

In this project, the project applied the source code analysis tool C Global Surveyor (http://ti.arc.nasa.gov/ase/cgs/), developed within the Automated Software Engineering group at NASA Ames under the Intelligent Systems program of Computing, Information, and Computing Technologies, to a payload software module for the International Space Station (ISS). The tool analyzes C programs to find dead code and memory access errors: de-references of null pointers and out-of-bounds array accesses, and in some cases uninitialized variables. The main benefits expected of applying the tool were finding errors in the software, validating the tool, and giving feedback to improve the tool.

The tool reports on the code by classifying operations as green, orange, or red. Green operations never result in a runtime error of the above types. Red operations always result in a runtime error. Orange operations are those for which the tool cannot determine one way or the other whether that operation would cause a runtime error (commonly referred to as "warnings"). An issue with such analysis tools is their scalability and the precision of their analysis. CGS was designed to run quickly on relatively large software and be precise about green operations; that is, it categorizes relatively few error-free operations as orange. It is probably less precise, though much faster, than Polyspace Verifier, another static analysis tool, about red operations; that is, operations that always cause errors might be classified by CGS as orange. The designers of CGS claim that its purpose is to do a complete coverage analysis of a software system to quickly narrow down the operations that need to be analyzed or tested further for whether they can cause an error. This follows because it was designed to be precise about which operations are green; thus, the amount of code for which further study is required will be minimized. The research infusion team had somewhat mischaracterized CGS's purpose as to flag errors in software, which requires the tool to be more precise about which operations are red. CGS had been applied to, and specialized in some ways for, Mars Pathfinder software and achieved 80% precision on it; that is, 80% of the operations were classified as red or green. This collaboration was something of an experiment to see if the tool could provide benefit for the analysis of other flight software.

The tool turned out to be about 50% precise on the module. If the tool were enhanced to deal with certain features of the C language and the application, the precision would have been about 90%. The project found its user interface cumbersome.

There were four important positive outcomes from the collaboration. First, dead code and an uninitialized variable were found in the module. Second, feedback was given to the CGS developers about new capabilities that the tool required to analyze certain features of C and handle this flight software. Third, serendipitously, because of his involvement in the collaboration, one of the CGS developers decided to apply another tool to the module which pinpointed a memory leak that had been suspected by the project. Lastly, as stated in the project's overview, "We continue to interface with the Technology Developers informally to derive Tool modifications, and to explore future uses of the tool for other collaborative efforts."

Success criteria 4 and 7 were met.

2.2 GSFC: "GSFC FSB Application of Perspective-Based Inspections"

The goal of this collaboration was to produce Flight Software Branch (FSB) process standards for software inspections which could be used across three new missions within the FSB. The standard was developed by Dr. Forrest Shull (Fraunhofer Center for Experimental Software Engineering, Maryland) using the Perspective-Based Inspection approach, (PBI research has been funded by SARP), then tested on a pilot Branch project. Because the short time scale of the collaboration ruled out a quantitative evaluation, it would be decided whether the standard was suitable for roll-out to other Branch projects based on a qualitative measure: whether the standard received high ratings from Branch personnel as to usability and overall satisfaction. The project used for piloting the Perspective-Based Inspection approach was a multi-mission framework designed for reuse. This was a good choice because key representatives from the three new missions would be involved in the inspections.

The perspective-based approach was applied to produce inspection procedures tailored for the specific quality needs of the branch. The technical information to do so was largely drawn through a series of interviews with Branch personnel. The framework team used the procedures to review requirements. The inspections were useful for indicating that a restructuring of the requirements document was needed, which led to changes in the development project plan.

The standard was sent out to other Branch personnel for review. Branch personnel were very positive. However, important changes were identified because the perspective of Attitude Control System (ACS) developers had not been adequately represented, a result of the specific personnel interviewed.

The net result is that with some further work to incorporate the ACS perspective, and in synchrony with the roll out of independent Branch standards, the PBI approach will be implemented in the FSB. Also, the project intends to continue its collaboration with the technology provider (Dr. Forrest Shull) past the end of the grant, to allow a more rigorous quantitative evaluation.

Success criteria 1, 3, 4, and 7 were met, and 2 and 5 are anticipated.

2.3 JPL: "Finding Defect Patterns in Reused Code" using Orthogonal Defect Classification

This effort used Orthogonal Defect Classification (ODC) to characterize defect reports for code that will be reused in mission-critical ground software. The application of ODC to NASA projects has been funded by SARP. The goal was to identify patterns of defects prior to reuse of the code, and to successfully infuse ODC into a project. ODC, as adapted for NASA by the researchers, characterizes anomaly reports using four attributes: Activity, Trigger, Target, and Type.

There were several groups of players in this project: Software Quality Assurance (SQA), JPL's Software Quality Initiative (SQI), Dr. Robyn Lutz (JPL, Iowa State University), and of course the ground software project. Dr. Lutz worked with the project to customize the classification entries. The original idea was to have the project itself learn to do the classification and analysis of anomaly reports on the software. However, the funding for the collaboration was late, the project entered a busy period, and there was a JPL reorganization, so instead people in SQA and SQI were taught the technique, and, with help from the project, classified the anomalies. Dr. Lutz did the analysis and reported the findings to the project. Infusing ODC into the SQA and SQI organizations was an unexpected benefit of the collaboration.

The project was satisfied with the results of the ODC analysis. Though the ground software project was not continued this year, so the software was not reused, the software whose anomalies were analyzed was recently put into operation, and the analysis results will be used to direct its maintenance. Another project by the same development team could use an ODC analysis. There are funds remaining, and that project will employ ODC. The project uses a bug tracking database that is compatible with ODC classifications. This will help introduce ODC because the classification can be done easily when the anomaly is reported, rather than later when it is more difficult to decipher the anomaly report.

Success criteria 1, 2, 3, 4, and 7 were met, and 5 is anticipated.

2.4 JSC: "Can CodeSurfer Increase Inspection Efficiency?"

CodeSurfer (http://www.grammatech.com/products/codesurfer/overview.html) is a commercial tool from Grammatech, Inc. for browsing C code. It provides lists of variables and constants used or set by functions, call graphs, pointer analysis, indications of dead code, etc. The objective of this project was for the Software Assurance organization to apply the tool during the inspection phase of an ISS software component, to see if the tool made the inspections more time efficient and/or more productive; that is, more defects found. Because the funding arrived late, and the acquisition took longer than anticipated, the window for the inspection phase of the module was missed. It was decided to apply CodeSurfer to the component anyway, as an experiment to compare

with previous inspection results. Also, CodeSurfer was applied during inspection of another ISS component.

The results show that the time required for doing an inspection using CodeSurfer is reduced from that for a manual inspection, and the inspection is more productive. Their final report states that manual code inspection required 17 hours, and only 12.25 hours with CodeSurfer. Manual code inspection found 8 defects, whereas 18 were found using CodeSurfer. Though the defects were all classified as minor, these are clear benefits. However there were difficulties. There is a learning curve: the training helped, but the project suggested that the tool would be difficult to use if there was a long time between uses, so ideally, there should be people who use the tool more frequently. The tool required that the code compile with one of the compilers provided with the tool: this ran into problems because the code analyzed would only compile using a legacy compiler, so some adaptation was required. Also, Software Assurance did not always readily have all the required files. The vendor of CodeSurfer, GrammaTech, Inc., was responsive, but because of ITAR restrictions, the ISS code could not be sent to the vendor for their assistance. The net effect was that setup time swamped inspection time. Obviously, there is a learning curve, so setup time would be reduced in the future. The research infusion team sees these as generic problems to be dealt with for code analysis tools.

The summary impact is that the Software Assurance organization plans to continue the use of CodeSurfer on C and C++ projects for reviews. They have demonstrated the tools to the engineers who developed the ISS components, and are interested in collaborating with other customers of Software Assurance in using the tool to troubleshoot software.

Success criteria 1, 2, 4, and 7 were met.

2.5 MSFC: "Static Analysis of Flight Software" using Coverity SWAT and C Global Surveyor

The objective of this effort was to apply two source code analysis tools to four flight software components, to find errors, and characterize the utility of the tools. The components varied in maturity from the coding and unit testing phase to the maintenance phase. The two analysis tools were C Global surveyor (characterized above in Section 3.1) and Coverity, Inc.'s Software Analysis Toolset (SWAT) (http://coverity.com/). The latter is a source code analysis tool for C programs that looks for certain types of errors, such as use of uninitialized variables, out-of-bounds indices (buffer overrun), dead code, and functions that should check their return value but don't. It does not claim complete coverage, in contrast to CGS, which does; that is, SWAT does not necessarily find all the errors of a particular type.

A team from MSFC was trained at ARC in the use of CGS. This resulted in a number of recommendations for the tool, similar to those found in the ARC collaboration. The tool produced about 300 warnings for a couple of the modules; about 20% were analyzed, and no errors were found. On the other hand, the technology developers reported that on one

of the MSFC applications, CGS was 95% precise. An update to CGS that fixed some of the issues raised was delivered to MSFC, but it was not run again on their software.

The Coverity tool was applied to the components. It flagged a total of 74 errors in 14 minutes. Analysis of those errors by flight engineers resulted in no errors found in the most mature component, but 9 in the other components were considered errors that were registered to be fixed; four of these had escaped formal testing. A usability issue was brought to the attention of Coverity.

The project concluded that the Coverity SWAT tool thus had a low false alarm rate and fast execution times and was recommended for use in the project's software development process if the associated licensing costs can be afforded.

Success criteria 1 (Coverity, not CGS), 4, 7, and a conditional 2 and 3 were met.

2.6 USA: "USA Application of Perspective-Based Inspections"

The Perspective-Based Inspection approach was applied by Dr. Forrest Shull in an ISS software development project at United Space Alliance. The goal was to increase the quality of the product, and increase inspection efficiency over previously used techniques. Project personnel were interviewed to tailor the approach, and instruction was provided, with actual software inspected as part of the instruction. Defects were found during that inspection, which was surprising because that software was reused from a previous version and hence thought to be defect free. After the course, Perspective-based inspections of code were carried out, finding a major defect which had escaped previous inspections. On a qualitative, subjective level, the response from the project team consisted of only positive comments.

The experience was that less time was required per inspector, who also had a more structured focus. It was noted that Perspective-Based Inspections required more inspectors than the project's usual practice. The approach will be recommended initially as an optional practice at USA. A kit was created to easily help craft perspectives for smaller projects. The project recommended the approach for larger projects.

Success criteria 1, 2, 3, 4, and 7 were met, and 5 is anticipated.

3 Paths to further infusion of the technologies

3.1 CGS

The purpose of the CGS tool has been clarified to be not simply finding errors, as with Coverity SWAT, so much as doing a full code coverage analysis that reduces the portions of the code that must be further analyzed or tested to ascertain whether a class of runtime errors are possible. This fits at the integration test phase of development as a certification tool, rather than unit test, where SWAT naturally fits.

Some users complained about the lack of a GUI, but the CGS developers feel that providing merely a color-coded presentation of results does not deal with the real issue of providing assistance for the analysis of, and/or construction of test cases for, orange operations (those for which a runtime error occurring was not ruled out). Funding to support development for this level of assistance is being sought--a possible SARP proposal.

The tool exploits features of the Mars Pathfinder/MER (Mars Exploration Rover) executive architecture; for example, it expects a limit on the depth of nested structures. A natural next application would be for the developers to run CGS on other JPL software derived from MER. However, JPL tends not to want to send their code out of JPL, for ITAR and other reasons. This is an issue because long visits to JPL by the CGS developers are not feasible.

The collaborations involving CGS were an experiment in applying the tool to other flight software at NASA; it was not known up front what the results would be. In the collaborations, the tool was imprecise because it needed to be extended to account for: the size of memory linked to hardware pointers; and handling bit fields. Also, the code structure of some of the applications was very different than the Mars Pathfinder/MER code structure, leading to imprecision.

3.2 Perspective-Based Inspections

Dr. Forrest Shull has developed a course syllabus for formal inspections. The syllabus includes tailoring for the attendees. The Knowledge and Training subgroup of the intercenter Software Working Group (SWG) is soliciting interest across NASA in the course, with the intent of funding it in FY05 if interest is sufficient.

At the moment, implementing Perspective-Based Inspections has a tailoring component. Dr. Shull expects that there is a limit to the number of perspectives that need to be produced for software. He provided a tailoring kit to USA.

3.3 ODC

SQI at JPL now includes ODC attributes among the list of metrics it recommends projects collect and analyze, and is willing to help projects interested in such metrics by providing modest support to get started at implementing their use.

The Reliability organization at JPL is rolling out a next generation anomaly reporting system called PRS (Problem Reporting System). It is being piloted on a flight project. It is extensible, built on a database. JPL is looking into which new fields and pull downs are needed so that PFRs (Problem Failure Reports) entered into PRS support ODC. The idea is to do an ODC analysis of Build 1 for the flight project to use in adjusting software

development processes for Build 2. If this is effective on the flight project, then PRS and ODC would be rolled out to other projects.

More generally, the Metrics subgroup of the SWG (Software Working Group) is working on tools to get metrics established on projects across NASA. One of their tools is a ProjectType-Goal-Metric matrix which is an aid to deciding which metrics should be collected to achieve a particular goal on a certain class of project. The research infusion team has been trying to connect the technology provider with the Metrics subgroup to see how ODC could be worked into the matrix, or find other avenues for infusion of ODC across NASA.

3.4 CodeSurfer

CodeSurfer seems to be reasonably priced; however, infrequent use would seem to be ineffective. It thus would be fit for use by software developers or certain software assurance personnel who would apply it more or less regularly. It does require compilation of code with one of their sets of compilers, which cannot always be done on legacy code. Another collaboration using this tool will take place at the IVVF, where it could have regular use if the collaboration is successful. CodeSurfer seems to exemplify a class of research products whose infusion may best be achieved by supporting a collaboration at each center, so as to allow each center the opportunity to "put its toe in the water". See the next section about an agency wide tools acquirer.

3.5 Coverity SWAT

Coverity SWAT seems effective, and attempts are being made to negotiate favorable licensing arrangements. JPL has a lab-wide tools acquirer that gets site licenses for tools, and then charges projects for their use. Could this work at other centers? Could an agency-wide tools acquirer be instituted?

4 Technology transfer lessons learned

1. Some developers are not proficient at research-oriented activities and need guidance and oversight. These teams are likely to benefit from more detailed *pro forma* documentation or templates (kick-off meeting agenda, project plan, reports). For specific categories of tools (such as source code analysis tools) we can provide very detailed templates. They also require frequent oversight (a) to be sure communication is occurring between developers and tech vendors and (b) to be sure that the schedule is being followed. Not all the projects require this level of support but it is likely to benefit Research Infusion by promoting uniform, higher-quality collaboration practice.

2. There are various answers to the question "What is the next step" – from research infusion to technology transfer. A general solution is unlikely. Some technologies

are readily integrated and generalized into a parent organization's existing processes (for example, Perspective-based Inspections at GSFC) – they are modifications to existing processes. Various other technology-specific approaches may be appropriate; e.g., PBI may be supported by the Software Engineering Initiative's Training strategy.

3. Tighter qualification of technology/project combination may be needed. One of the source code analysis tools used at ARC and MSFC had previously been successfully applied to NASA software, but software that had different technical features. The tool did not transition well to software that did not have these features. Also, the appropriate lifecycle context and purpose for the tool (in this case) may not have been clear to the development teams.

4. Collaborations' project plans should explicitly include an iterative approach to technology application, scaling up with each iteration, as cited in the GSFC and JPL collaborations' final reports.

5. To succeed, training and continued support are needed. For example, USA received onsite training on applying PBI technology to its own application. This reduced risk and cost as well, since part of the target application was used in the training session. "The most successful way to do tech transfer is to put a member of the [technology vendor team] on the development team" – Matt Barry, JPL, (paraphrased) communication to the authors.

6. Overall, Research Infusion's first set of completed collaborations supports the hypothesis that with selection of appropriate technologies, matching of technology with software development team, and guidance and oversight, infusion of new software engineering technologies can be performed successfully on a minimal budget.

5 Acknowledgements

This report incorporates material and suggestions from the following individuals.

Research Infusion Team

Benedetto Di Vito, LaRC
Martin Feather, JPL
Michael Hinchey, GSFC
Luis Trevino, MSFC

Project Members and Technology Providers (TP)

Susanne Moran, Intrinsyx (ARC)
Guillaume Brat, RIACS (ARC) (TP)

Arnaud Venet, Kestrel Technology (ARC) (TP)
Elaine Shell, GSFC
Michael Stark, GSFC
Michael Tilley, Raytheon (GSFC)
Forrest Shull, Fraunhofer Center for Experimental Software Engineering, Maryland (TP)
Scott Morgan, JPL
Martha Berg Strain, JPL
Steve Rockwell, JPL
Tuan Do, JPL
Robyn Lutz, JPL & Iowa State University (TP)
Carmen Mikulski, JPL (TP)
Mark Markovich, SAIC (JSC)
Dan Freund, JSC
Carl Soderland, JSC
Tara Ruttley, JSC
Scott Akridge, MSFC
Jerry Crook, Lockheed Martin (MSFC)
Justin Thomas, USA

6 Acronyms

ACS: Attitude Control System
ARC: NASA Ames Research Center
CGS: C Global Surveyor, a static analysis tool for C software developed at NASA Ames.
FSB: Flight Software Branch (GSFC)
FSW: Flight Software
GSFC: NASA Goddard Space Flight Center
ISS: International Space Station
ITAR: International Traffic in Arms Regulations
IVVF: NASA Independent Verification and Validation Facility (West Virginia)
JPL: NASA Jet Propulsion Laboratory
JSC: NASA Johnson Space Center
LaRC: NASA Langley Research Center
MER: Mars Exploration Rover
MSFC: NASA Marshall Space Flight Center
NASA: National Aeronautics and Space Administration
ODC: Orthogonal Defect Classification
PBI: Perspective-Based Inspections
PF: Penetration Factor
PFR: Problem Failure Report
PRS: Problem Reporting System
SARP: Software Assurance Research Program (NASA Office of Safety and Mission Assurance)
SQA: Software Quality Assurance
SQI: Software Quality Initiative (JPL)

SWAT: SoftWare Analysis Toolset (from Coverity, Inc.)
SWG: Software Working Group
TRL: Technology Readiness Level
USA: United Space Alliance

7 References

7.1 Collaboration Reports

"On-Orbit Software Analysis", NASA ARC, Susanne Moran, POC.

"GSFC FSB Application of Perspective-Based Inspections," NASA GSFC, Elaine Shell, POC.

"Research Infusion Collaboration: Finding Defect Patterns in Reused Code," NASA JPL, Robyn Lutz and Scott Morgan, with contributors Tuan Do, Carmen Mikulski, Martha Berg Strain, Steve Rockwell, and Belinda Wilkinson.

"Can CodeSurfer Increase Code Inspection Efficiency?", NASA JSC, Mark Markovich, POC.

"Static Analysis of Flight Software," NASA MSFC, Scott Akridge, POC.

"USA Application of Perspective-Based Inspections," United Space Alliance, Justin Thomas, POC.

7.2 Technologies

Technologies deployed in 2004 are described at http://ic.arc.nasa.gov/researchinfusion/materials/2004/index.php . There are also the following references.

7.2.1 CGS

See http://ti.arc.nasa.gov/ase/cgs/index.html

Arnaud Venet and Guillaume Brat, "Precise and Efficient Static Array Bound Checking for Large Embedded C Programs," Proceedings of the International Conference on Programming Language Design and Implementation (PLDI'04), Washington DC, USA , pp. 231-242, ACM Press 2004.

Guillaume Brat and Arnaud Venet, "Precise and Scalable Static Program Analysis of NASA Flight Software," Proceedings of the 2005 IEEE Aerospace Conference, Big Sky, MT, USA, March 5-12, 2005, IEEE Press 2005.

7.2.2 Perspective-Based Inspections

Shull F., Rus I., and Basili V. R., "How Perspective-Based Reading Can Improve Requirements Inspections," IEEE Computer, vol. 33, no. 7, pp. 73-79, July 2000.

Travassos G. H., Shull F., Fredericks M., and Basili V. R., "Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality", In Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), November 1999.

Basili, V., Green , S., Laitenberger, O., Lanubile, F., Shull, F., Soerumgaard, S., and Zelkowitz, M. "The Empirical Investigation of Perspective-Based Reading." Empirical Software Engineering: An International Journal, 1(2): 133-164, 1996

7.2.3 Orthogonal Defect Classification

See Day One, Session 4 at http://sas.ivv.nasa.gov/conclusion2003.html .

See http://sarpresults.ivv.nasa.gov/ViewResearch/165/13.jsp .

Robyn Lutz and Carmen Mikulski, "Empirical Analysis of Safety-Critical Anomalies during Operations," IEEE Transactions on Software Engineering, vol. 30, no. 3, March, 2004, pp. 172-180. (The most useful for guiding process improvement.)

Robyn Lutz and Carmen Mikulski, "On-Going Requirements Discovery in High-Integrity Systems," IEEE Software, vol. 21, no. 2, March-April, 2004, pp. 19-25. (The most interesting to developers thinking of using ODC.)

Robyn Lutz and Carmen Mikulski, "Resolving Requirements Discovery in Testing and Operations," Proceedings of the 11th IEEE Requirements Engineering Conference (RE'03), Sept. 8-12, 2003, Monterey Bay, CA, pp. 33-41.

Robyn Lutz, Tim Menzies, and Carmen Mikulski, "Better Analysis of Defect Data at NASA," Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'03), July 1-3, 2003, San Francisco, CA.

Robyn Lutz and Carmen Mikulski, "Patterns of Software Defect Data on Spacecraft," International Conference on Space Mission Challenges for Information Technology (SMC-IT'03), Pasadena, CA, July 13-16, 2003.

Robyn Lutz and Carmen Mikulski, "Requirements Discovery during the Testing of Safety-Critical Software," Proceedings of the 25th International Conference on Software Engineering (ICSE'03), May 3-10, 2003, Portland, OR, pp. 578-583.

7.2.4 CodeSurfer

See http://www.grammatech.com/products/codesurfer/overview.html .

7.2.5 Coverity SWAT

The new name for SWAT is Prevent. See http://coverity.com/ .